

ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU



VEB PORTAL ZA OBJAVLJIVANJE BLOGOVA

– Master rad –

Kandidat:

Luka Tadić 2016/3260

Mentor:

vanr.prof. dr Zoran Čiča

Beograd, April 2018.

SADRŽAJ

SADRŽAJ	2
1. UVOD	3
2. OSNOVNI ALATI I PROGRAMSKI JEZICI	4
2.1. HTML.....	4
2.2. CSS.....	4
2.3. JAVASCRIPT.....	4
2.3.1. <i>jQuery</i>	5
2.4. PHP.....	5
2.5. MYSQL.....	5
2.6. XAMPP.....	5
3. UPUTSTVO ZA KORIŠĆENJE APLIKACIJE	6
3.1. UPUTSTVO ZA OBIČNOG KORISNIKA.....	6
3.1.1. <i>Naslovna strana</i>	6
3.1.2. <i>Pretraga i napredna pretraga</i>	7
3.1.3. <i>Stranica „Blogeri“</i>	8
3.1.4. <i>Stranica „Moji podaci“</i>	8
3.1.5. <i>Stranice „About“ i „Contact“</i>	9
3.1.6. <i>Post</i>	10
3.1.7. <i>Stranica kategorije</i>	11
3.2. UPUTSTVO ZA BLOGERA.....	11
3.2.1. <i>Dodavanje posta</i>	11
3.2.2. <i>Ažuriranje posta</i>	12
3.3. UPUTSTVO ZA ADMINISTRATORA.....	13
3.3.1. <i>Upravljanje postovima</i>	13
3.3.2. <i>Upravljanje korisnicima</i>	14
4. OPIS IMPLEMENTACIJE	16
4.1. STRUKTURA BAZE PODATAKA.....	16
4.2. NAJVAŽNIJI DELOVI KODA.....	17
4.2.1. <i>Struktura projekta</i>	17
4.2.2. <i>Povezivanje sa bazom podataka i generisanje odgovora za klijenta</i>	18
4.2.3. <i>Realizacija pretrage</i>	19
4.2.4. <i>Realizacija dodavanja posta</i>	27
4.2.5. <i>Prikaz posta</i>	30
5. ZAKLJUČAK	36
LITERATURA	37

1. UVOD

Veb portal označava specijalno dizajniranu veb aplikaciju koja često služi kao jedinstvena tačka pristupa informacijama. Takođe se može smatrati bibliotekom personalizovanog i kategorizovanog sadržaja. Veb portali registrovanim korisnicima pružaju mogućnost da utiču na izgled i na sadržaj samog portala. Korisnicima se pruža mogućnost pretrage i izdvajanja informacija koje su njima zanimljive.

Veb aplikacija realizovana u okviru teze posvećena je automatizaciji procesa vođenja bloga. U zavisnosti od tipa kome pripadaju, korisnici će moći da vrše pregled i pretragu sadržaja, dodavanje novog, kao i ažuriranje i brisanje već postojećeg sadržaja.

U narednom poglavlju biće reči o softverskim alatima i programskim jezicima koji su korišćeni u okviru teze. U trećem poglavlju je predstavljeno detaljno uputstvo za korišćenje aplikacije sa stanovišta korisnika i sa stanovišta administratora. Četvrto poglavlje je detaljan opis važnijih delova koda, dok peto poglavlje čini zaključak u kome će se rezimirati rezultati teze, potencijalna poboljšanja i primene.

2. OSNOVNI ALATI I PROGRAMSKI JEZICI

Programski jezici i alati koji su korišćeni za realizaciju klijentskog dela veb aplikacije su *HTML*, *CSS*, *JavaScript*, *jQuery*, dok su za serverski deo korišćeni *PHP*, *MySQL* i *XAMPP*.

2.1. HTML

HTML (*HyperText Markup Language*) je veoma jednostavan jezik koji služi za opis veb stranica. Osnovu jezika HTML predstavljaju tagovi i atributi. Tagovima se određeni deo dokumenta odvaja od ostatka i na njega se primenjuju pravila definisana samim tagom. Atributi se nalaze unutar tagova i omogućavaju da se pored samog imena taga i unapred definisanog ponašanja, još bliže odredi način prikaza i ponašanja označenog dela dokumenta. [8]

HTML stranice imaju ekstenziju *.html* ili *.htm* i nalaze se u određenom direktorijumu servera povezanog na Internet, što ih čini dostupnim na vebu. Pomoću HTML jezika se generišu dokumenti tipa hipertekst. [1]

Hipertekst je tekst koji sadrži veze ili linkove ka drugim dokumentima ili ka samom sebi. To je skup stranica međusobno povezanih linkovima koji su umetnuti u stranice. Za razliku od običnog teksta koji se čita linearno (sa leva na desno, odozgo naniže), hipertekst se čita prateći hiper-veze u tekstu, dakle, ne nužno na linearan način. [1]

2.2. CSS

CSS (*Cascading Style Sheets*) je jezik koji omogućava formatiranje i opis izgleda dokumenta napisanih HTML jezikom. HTML bi trebalo da se koristi za opis strukture dokumenta, dok se vizuelna definicija HTML stranica prepušta stilovima koji se definišu pomoću jezika CSS. Stilovi se definišu za HTML tagove i oni određuju poziciju i izgled taga (boja, font, pozadinska boja, itd.).

CSS stilovi se ugrađuju na tri standardna mesta: unutar samog HTML taga, koristeći atribut *style* HTML taga i kreiranjem eksterne stranice stilova (*.css* fajl) koji se u dokument uključuje tagom *<link>*.

Dakle, glavne prednosti koje donosi CSS su bolja kontrola izgleda stranice, razdvajanje sadržaja i formatiranja, brže učitavanje stranica, kao i lakše održavanje i ažuriranje više stranica istovremeno.

2.3. JavaScript

JavaScript je skriptni programski jezik koji se prvenstveno koristi za definisanje funkcionalnosti veb stranica na klijentskoj strani. Pripada grupi skript jezika jer se sastoji od serije komandi koje se očitavaju u interpreteru, bez prethodnog kompajliranja sadržaja. Izvršava se na strani korisnika, odnosno na računaru na kojem je pokrenut sadržaj sa *JavaScript* kodom.

JavaScript može menjati sadržaj prikazane veb stranice i može kontrolisati brauzer. Omogućava dinamički HTML sadržaj, ima mogućnost da menja vrednosti HTML tagova i atributa.

Reaguje na događaje tipa klika, slanja formulara, itd. Ostvaruje razne vremenske funkcije i koristi se za validaciju podataka unetih od strane korisnika pre njihovog slanja na server, na obradu. [1]

2.3.1. jQuery

jQuery je biblioteka koja pojednostavljuje korišćenje jezika *JavaScript* kod zahtevnih funkcionalnosti kao što su *AJAX (Asynchronous JavaScript and XML)* pozivi ili manipulacija nad *DOM (Document Object Model)* objektom.

2.4. PHP

PHP (Hypertext Preprocessor) je skriptni jezik koji se izvršava na strani servera i pre svega je namenjen za izradu dinamičkog veb sadržaja. Može da memoriše podatke i skladišti ih, pa ih koristi kasnije u nekom trenutku. *PHP* je platformski neutralan jezik i sličan je programskom jeziku *C* od koga je i nastao. Omogućava dinamički *HTML* sadržaj. Vršiti obradu podataka i omogućava povezivanje sa bazama podataka na serveru. Implementira autentifikaciju i zaštitu komunikacije. Svaka dinamička veb stranica koja u svom kodu ima neku *PHP* skriptu koja se izvršava treba da ima ekstenziju *.php* kako bi veb server znao da postoji *PHP* kod koji treba da se izvrši.

2.5. MySQL

MySQL (My Structured Query Language) je najpopularniji sistem otvorenog koda za upravljanje bazama podataka. Zasnovan je na *SQL (Structured Query Language)* jeziku namenjenom za pristupanje i manipulisanje bazama podataka.

MySQL je višenitni, višekorisnički sistem za upravljanje relacionim bazama podataka. U relacionoj bazi podataka se podaci smeštaju u više međusobno povezanih tabela. Ovim se dobija na brzini i fleksibilnosti. Korišćenje ove baze podataka je besplatno.

MySQL komande nisu osetljive na veličinu slova, ali je običaj da se pišu velikim slovima. Promenljive jesu osetljive na veličinu slova. *MySQL* nam omogućava da stvorimo i promenimo strukturu baze podataka, dodamo prava korisniku za pristup bazi podataka, da tražimo informacije od baze podataka i menjamo njenu strukturu. [9]

2.6. XAMPP

XAMPP (Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) and Perl (P)) je softverski paket koji omogućava korisniku da na sopstvenom računaru simulira rad veb servera. *Apache* veb server omogućava da se na korisnikovom računaru istovremeno nalazi i korisnik i veb server.

U paket mogu biti uključeni i dodatni programi, kao, na primer, *phpMyAdmin*, kreiran u *PHP*-u, koji služi za kreiranje, modifikovanje i brisanje baze podataka, kolona, tabela ili drugih oblika podataka koristeći *SQL* programski jezik.

3. UPUTSTVO ZA KORIŠĆENJE APLIKACIJE

Prema privilegijama pristupa određenim funkcionalnostima veb aplikacije, definisane su tri grupe korisnika:

- Običan korisnik
- Blogger
- Admin

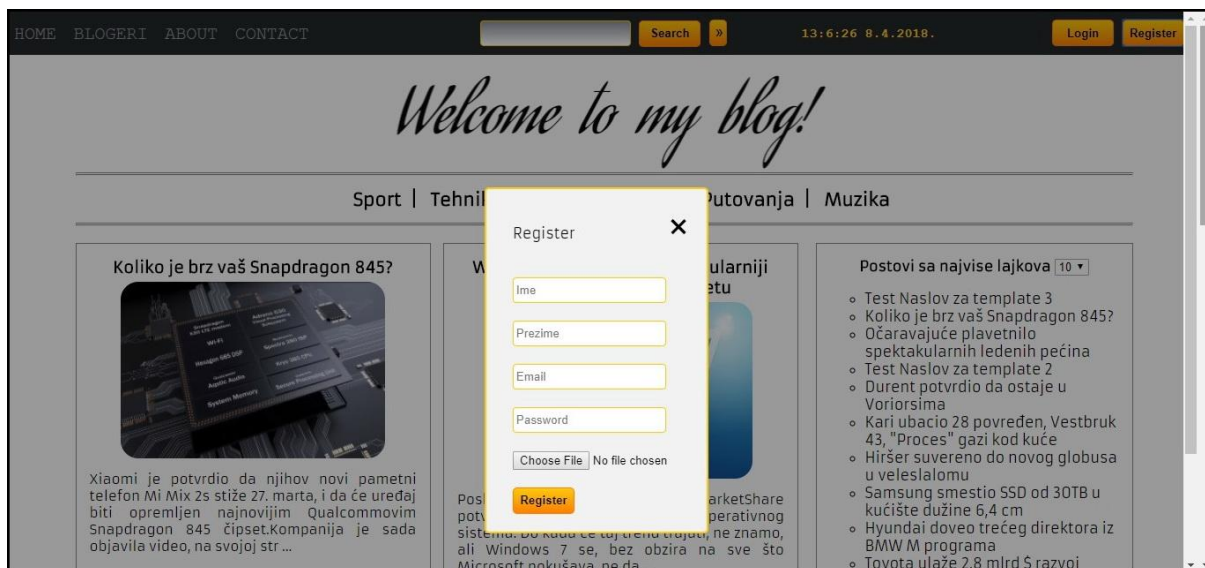
U skladu sa tim, u daljem tekstu biće izloženo uputstvo za korišćenje aplikacije sa stanovišta svakog od navedenih tipova korisnika.

3.1. Uputstvo za običnog korisnika

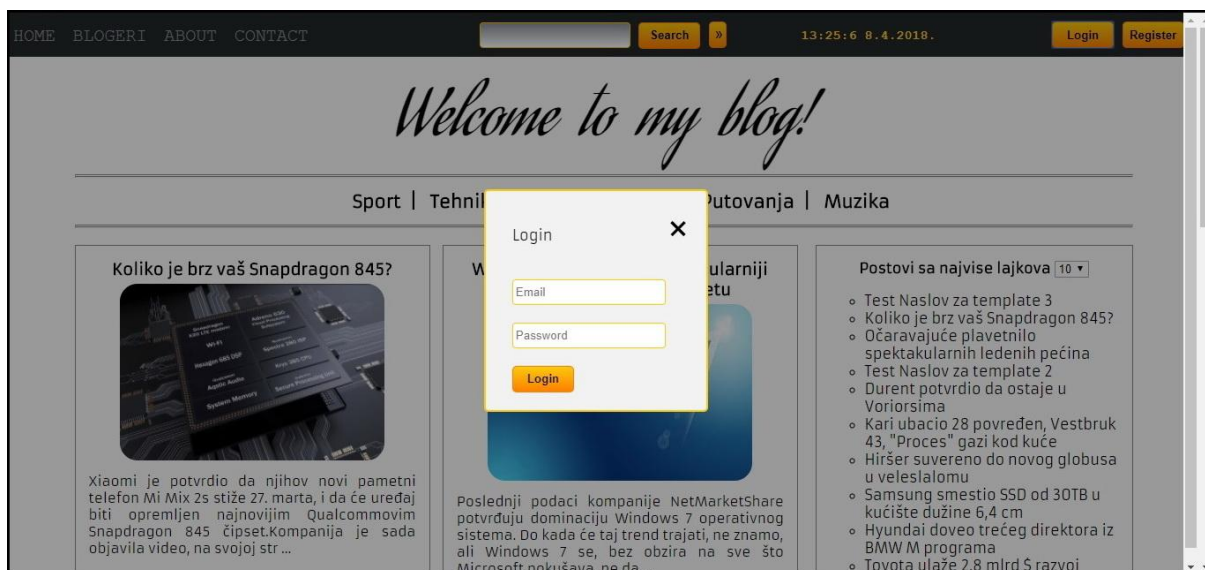
3.1.1. Naslovna strana

Na naslovnoj strani se nalaze linkovi za pristup osnovnim funkcionalnostima aplikacije. Vizuelno je podeljena u nekoliko celina.

Pre svega, da bi korisnik mogao da ima bilo kakvu interakciju sa aplikacijom potrebno je da se registruje i uloguje. Na vrhu naslovne strane, u desnom delu navigacionog menija, nalaze se dugmići „Register“ i „Login“. Klikom na navedene dugmiće otvaraju se modalni prozori sa formularima za unos podataka što je prikazano na slikama 3.1.1.1. i 3.1.1.2.



Slika 3.1.1.1. Izgled modalnog prozora za registraciju.



Slika 3.1.1.2. Izgled modalnog prozora za logovanje.

Nakon uspešne registracije i logovanja, korisniku su na raspolaganju sve opcije iz navigacionog menija. U levom delu navigacionog menija nalaze se linkovi ka stranicama „HOME“, „BLOGERI“, „MANAGE“, „ABOUT“ i „CONTACT“. Osim navedenih linkova za kretanje kroz aplikaciju, u meniju se nalazi deo vezan za pretragu, trenutni datum i vreme, kao i dugme „Logout“.

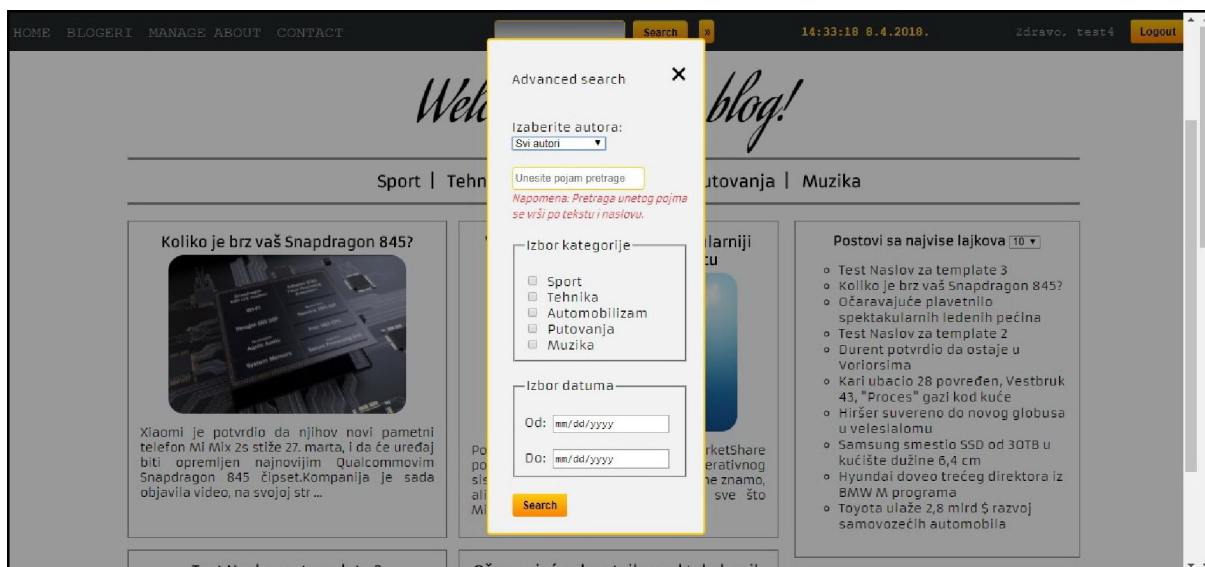
Ispod navigacionog menija nalazi se prikaz kategorija. Klikom na odgovarajuću kategoriju otvara se stranica sa svim postovima iz date kategorije.

U središnjem delu naslovne strane prikazani su postovi sortirani po datumu unosa. Korisnik, ispod poslednjeg posta u padajućem meniju, ima opciju za izbor prikaza broja postova na naslovnoj strani. Uz desnu ivicu naslovne strane, korisnik ima pregled postova grupisanih po broju lajkova i komentara, kao i prikaz najaktivnijih blogera. U dnu naslovne strane nalazi se još i futer.

3.1.2. Pretraga i napredna pretraga

U središnjem delu navigacionog menija nalaze se elementi za pretragu. Unošenjem željenog pojma u tekstualno polje i klikom na dugme „Search“ vrši se pretraga po naslovu i tekstu postova.

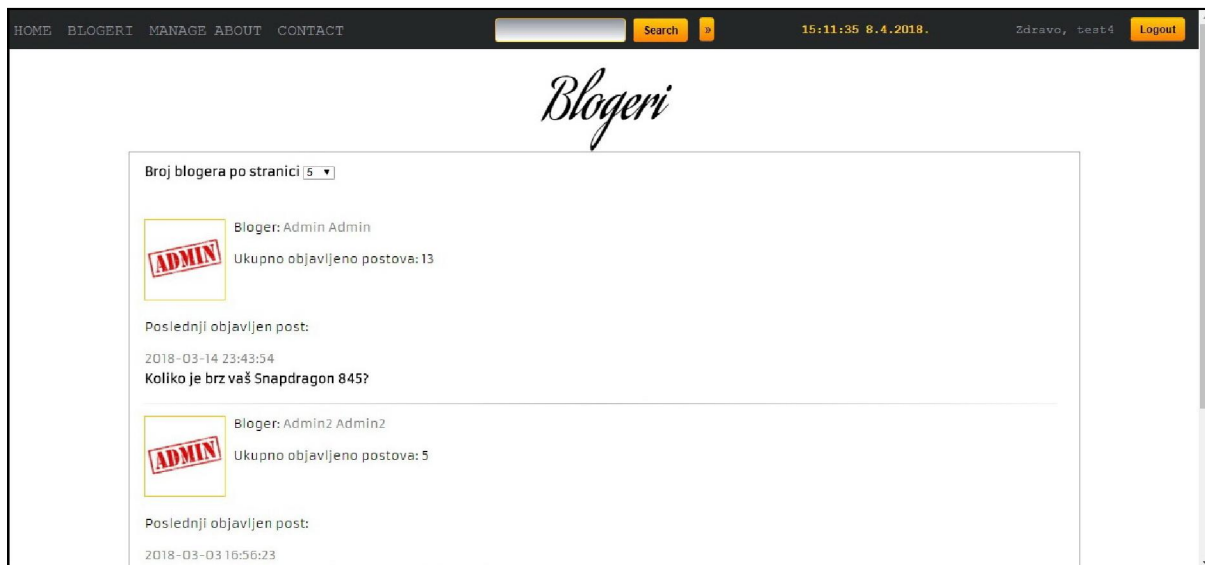
Pored dugmeta „Search“, nalazi se dugme sa simbolom „»“. Klikom na pomenuto dugme otvara se modalni prozor za naprednu pretragu. U naprednoj pretrazi, korisnik pored željenog pojma, ima mogućnost izbora autora, jedne ili više kategorija, kao i opseg datuma za pretragu. Izgled modalnog prozora za naprednu pretragu prikazan je na slici 3.1.2.1. Klikom na dugme „Search“ i potvrdom unesenih parametara za pretragu, osnovnu ili naprednu, otvara se stranica sa rezultatima pretrage.



Slika 3.1.2.1. Izgled modalnog prozora za naprednu pretragu.

3.1.3. Stranica „Blogeri“

Klikom na link „Blogeri“ u navigacionom meniju, otvara se odgovarajuća stranica na kojoj su izlistani autori sa podacima o broju objavljenih postova, kao i osnovne informacije o poslednjem objavljenom postu. Na vrhu ove stranice, korisnik ima mogućnost izbora broja autora koji će se prikazivati na jednoj stranici. Izgled ove stranice prikazan je na slici 3.1.3.1.

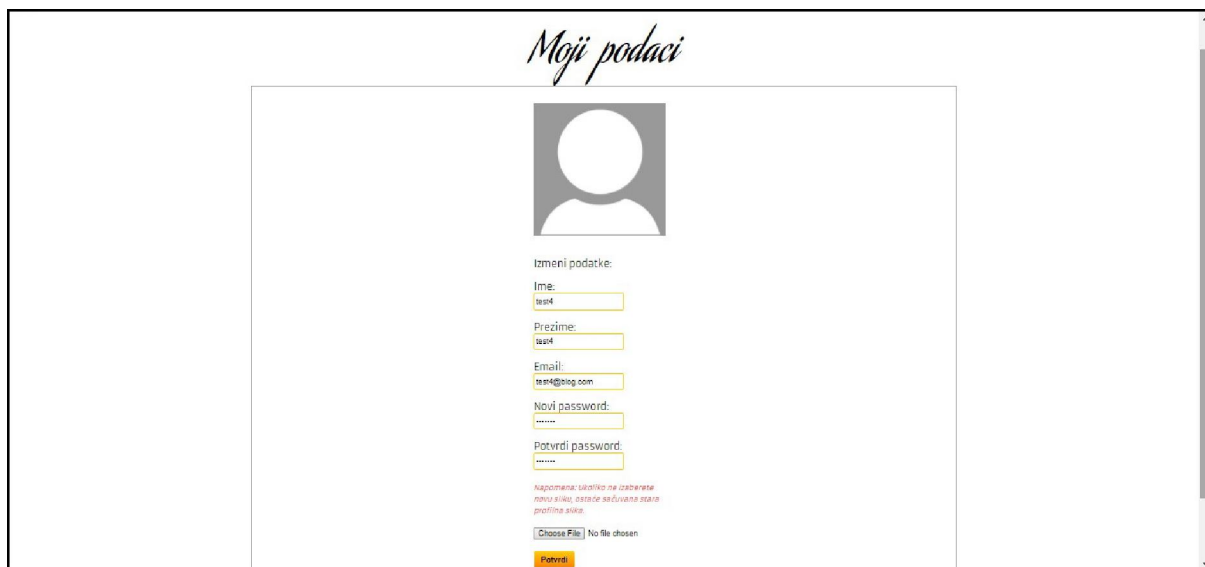


Slika 3.1.3.1. Izgled stranice „Blogeri“.

3.1.4. Stranica „Moji podaci“

Izborom podopcije „Moji podaci“, u okviru opcije „MANAGE“ u navigacionom meniju, otvara se stranica na kojoj su prikazani podaci ulogovanog korisnika. Izmenom odgovarajućih tekstualnih polja ili aploudovanjem nove slike, korisnik ima opciju da izmeni svoje postojeće podatke. Ukoliko korisnik želi da sačuva izmene potrebno je da to potvrdi klikom na dugme

„Potvrdi“, smešteno ispod formulara za unos podataka. Izgled formulara za izmenu podataka prikazan je na slici 3.1.4.1.

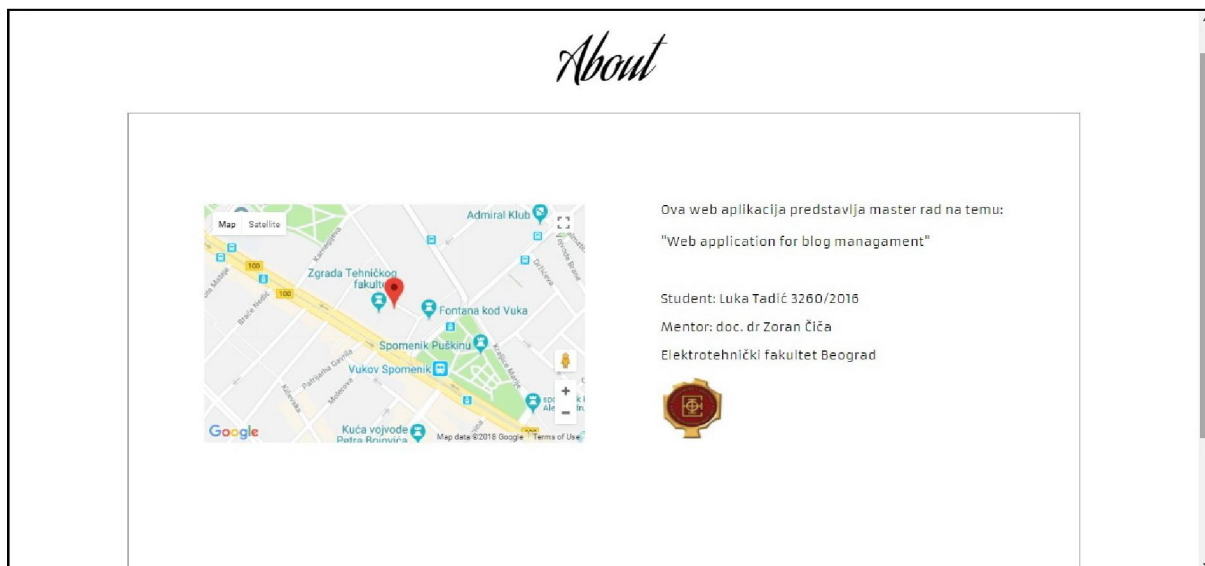


The screenshot shows a web page titled "Moji podaci" in a cursive font. Below the title is a placeholder for a profile picture. Underneath is a section labeled "Izmeni podatke:" (Edit data:). This section contains several input fields: "Ime:" (Name) with the value "test4", "Prezime:" (Surname) with "test4", "Email:" with "test4@eag.com", "Novi password:" (New password) with ".....", and "Potvrni password:" (Confirm password) with ".....". Below these fields is a red note: "Napomena: uvek treba izabrati novu sliku, ostale se čuvaju stare profilne slike." (Note: you always have to choose a new picture, the others are saved as old profile pictures). At the bottom of the form are two buttons: "Choose File" (disabled, showing "No file chosen") and "Potvrdi" (Confirm).

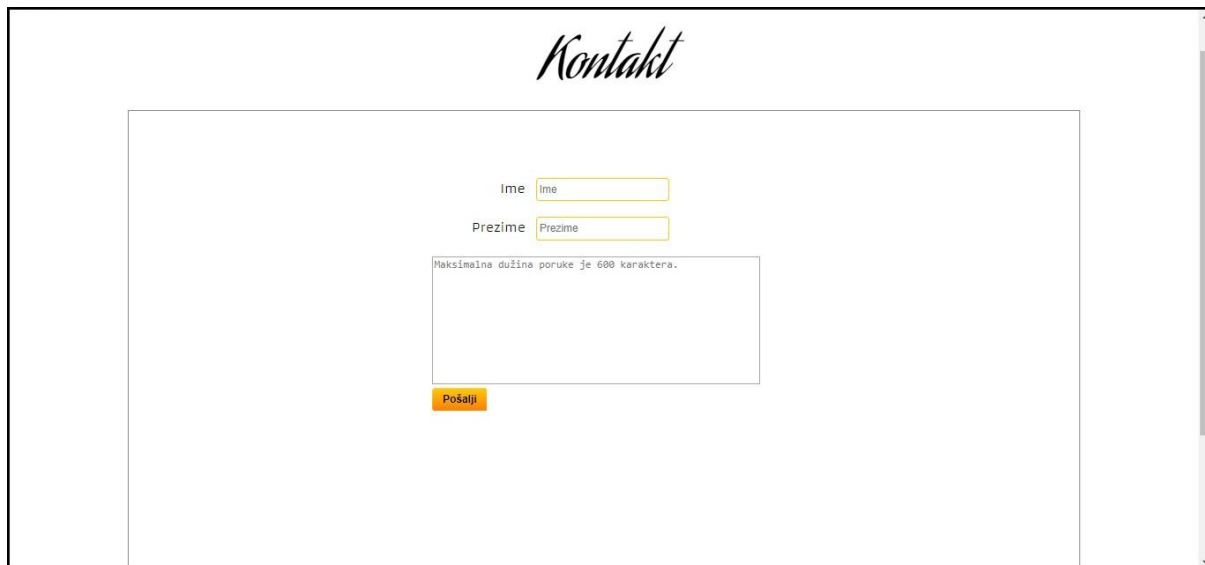
Slika 3.1.4.1. Izgled formulara za izmenu korisničkih podataka.

3.1.5. Stranice „About“ i „Contact“

Izborom odgovarajućih linkova u navigacionom meniju, otvaraju se stranice „About“ i „Contact“. Na „About“ stranici se mogu naći opšti podaci o veb aplikaciji, dok se na stranici „Contact“ nalazi formular za slanje email poruke administratoru veb aplikacije. Izgled ovih stranica prikazan je na slikama 3.1.5.1. i 3.1.5.2.



Slika 3.1.5.1. Izgled stranice „About“.



Slika 3.1.5.2. Izgled stranice „Contact“.

3.1.6. Post

Dolaskom na stranicu odgovarajućeg posta, u zavisnosti od templejta, prikazane su slike i tekst raspoređen u jednom ili dva odeljka. Sa desne strane, nalaze se izlistani naslovi povezanih postova tj. postova koji pripadaju istoj kategoriji kao i post na čijoj se stranici korisnik trenutno nalazi.

Ispod podataka o samom postu, nalaze se odgovarajući dugmići za lajk odnosno dislajk. Pored toga, popunjavanjem odgovarajućeg formulara, korisnik ima mogućnost dodavanja komentara za trenutni post. Takođe, postoji i opcija za prikaz svih komentara za dati post. Pomenute opcije imaju samo ulogovani korisnici, dok će korisnici koji ne zadovoljavaju ovaj uslov, pri pokušaju neke od navedenih opcija dobiti odgovarajuće obaveštenje. Izgled posta je prikazan na slici 3.1.6.1.



Slika 3.1.6.1. Izgled posta.

3.1.7. Stranica kategorije

Izborom odgovarajuće kategorije, otvara se stranica sa prikazom svih postova iz izabrane kategorije. Specifičnost ove stranice, jeste u tome što korisnik ima izbor za sortiranje prikaza. Na raspolaganju je opcija za izbor prikaza broja postova po stranici, kao i mogućnost sortiranja postova po datumu, broju lajkova i broju komentara. Izgled stranice kategorije prikazan je na slici 3.1.7.1.



Slika 3.1.7.1. Izgled stranice kategorije.

3.2. Uputstvo za blogera

Pored funkcionalnosti koje su opisane u odeljku 3.1, blogeru su na raspolaganju još dve bitne opcije koje ga u suštini i čine blogerom. To su dodavanje novih i ažuriranje postojećih postova.

3.2.1. Dodavanje posta

Izborom podopcije „Dodaj post“, u okviru opcije „MANAGE“ u navigacionom meniju, otvara se stranica za dodavanje posta. Korisnik ima na raspolaganju tri templejta sa različitim rasporedom i prikazom teksta i slika. U svakom templejtu se nalaze polja koja su neophodna za dodavanje posta, kao što su padajući meni za izbor kategorije, naslov, polja za unos teksta i odgovarajući dugmići za aploudovanje slika. Pozicije ovih polja su raspoređene onako kako bi izgledao post ukoliko se korisnik odluči za taj templejt. Klikom na dugme „Potvrdi“ otvara se stranica sa novounetim postom. Izgled jednog templejta prikazan je na slici 3.2.1.1.

Slika 3.2.1.1. Izgled templejta za dodavanje posta.

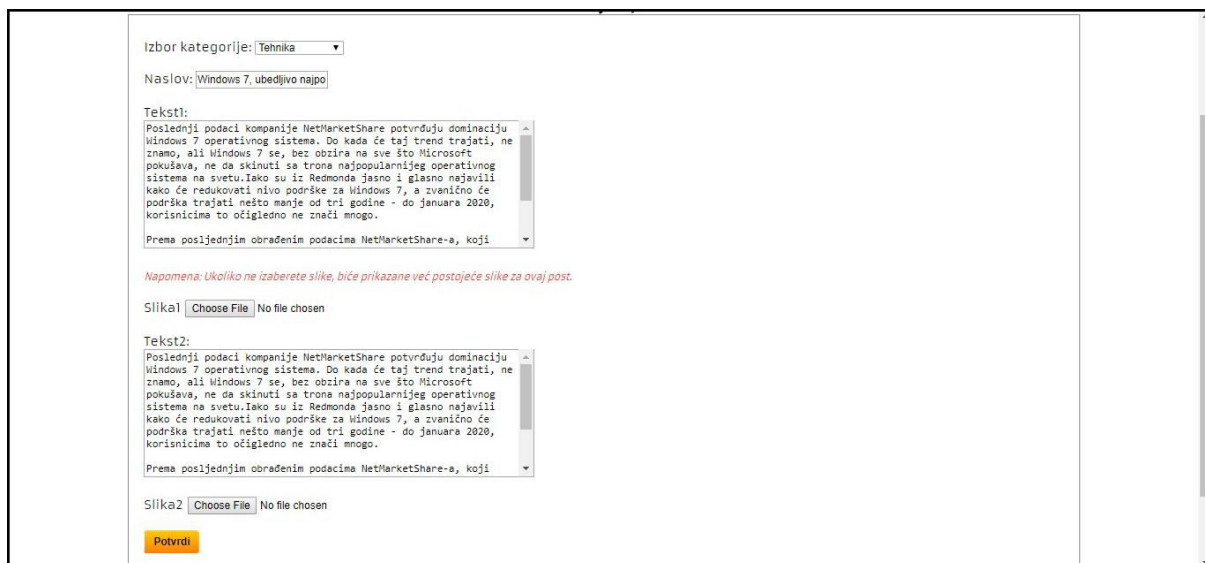
3.2.2. Ažuriranje posta

Bloger u svakom trenutku ima opciju ažuriranja i brisanja svojih postova. Izborom podopcije „Ažuriranje“, u okviru opcije „MANAGE“ u navigacionom meniju, otvara se stranica sa izlistanim postovima koje je uneo ulogovani korisnik. Na vrhu stranice postoji opcija za izbor prikaza broja postova po stranici. Izgled stranice prikazan je na slici 3.2.2.1.

Slika 3.2.2.1. Izgled stranice za ažuriranje postova.

Za svaki post su prikazani osnovni podaci kao što su naslov, datum objave, slika, početni deo teksta, kao i broj lajkova i dislaikova. Pored ovih podataka, ispod svakog posta nalaze se dugmići „Ažuriraj“ i „Obriši“. Klikom na dugme „Obriši“ otvara se prozor u kome se traži potvrda korisnika za brisanje datog posta. Klikom na dugme „Ažuriraj“ otvara se stranica identična onoj za unos

posta. Razlika jeste u tome što se sada otvara baš onaj templejt za koji je dati post definisan. Sada su odgovarajuća polja za naslov i tekst popunjena postojećim podacima iz baze podataka i korisnik ima mogućnost ažurira dati post. Klikom na dugme „Potvrdi“ otvara se stranica sa ažuriranim postom. Izgled formulara za ažuriranje odgovarajućeg posta prikazan je na slici 3.2.2.2.



Izbor kategorije: Tehnika

Naslov: Windows 7, ubedljivo najpo

Tekst1:
Poslednji podaci kompanije NetMarketShare potvrđuju dominaciju Windows 7 operativnog sistema. Do kada će taj trend trajati, ne znamo, ali Windows 7 se, bez obzira na sve što Microsoft pokušava, ne da skinuti sa trona najpopularnijeg operativnog sistema na svetu. Iako su iz Redmonda jasno i glasno najavili kako će redukovati nivo podrške za Windows 7, a zvanično će podrška trajati nešto manje od tri godine - do januara 2020, korisnicima to očigledno ne znači mnogo.
Prema posljednjim obrađenim podacima NetMarketShare-a, koji

Napomena: Ukoliko ne izaberete slike, biće prikazane već postojeće slike za ovaj post.

Slika1 No file chosen

Tekst2:
Poslednji podaci kompanije NetMarketShare potvrđuju dominaciju Windows 7 operativnog sistema. Do kada će taj trend trajati, ne znamo, ali Windows 7 se, bez obzira na sve što Microsoft pokušava, ne da skinuti sa trona najpopularnijeg operativnog sistema na svetu. Iako su iz Redmonda jasno i glasno najavili kako će redukovati nivo podrške za Windows 7, a zvanično će podrška trajati nešto manje od tri godine - do januara 2020, korisnicima to očigledno ne znači mnogo.
Prema posljednjim obrađenim podacima NetMarketShare-a, koji

Slika2 No file chosen

Slika 3.2.2.2. Izgled formulara za ažuriranje posta.

3.3. Uputstvo za administratora

Pored funkcionalnosti opisanih u odeljcima 3.1 i 3.2, administrator (u nastavku admin) ima i dodatne opcije na raspolaganju. U aplikaciji trenutno postoje definisana dva admin naloga sa sledećim podacima:

- 1) Email: admin@blog.com, Šifra: admin,
- 2) Email: admin2@blog.com, Šifra: admin2.

3.3.1. Upravljanje postovima

Za razliku od blogera, izborom stranice za ažuriranje postova, adminu se izlistavaju i postovi koje je on uneo, ali i postovi svih blogera. Izgled same stranice je identičan onom koji je prikazan na slici 3.2.2.1. Od svih postojećih postova u bazi podataka, admin neće imati na raspolaganju za ažuriranje i brisanje samo postove drugog admina. Na ovaj način je postignuto da pravo na menjanje posta koji je unet od strane admina ima jedino admin koji je post i uneo.

Radi veće fleksibilnosti u upravljanju postovima admin ima još jednu važnu funkcionalnost. Prilikom pregleda odgovarajućeg posta, u levom uglu iznad naslova, nalaze se dugmići „Ažuriraj“ i „Obriši“. Ovi dugmići imaju istu funkcionalnost kao i na stranici za ažuriranje postova. Takođe, kao i na stranici za ažuriranje postova, admin nema mogućnost da menja postove drugog admina. Ukoliko admin pregleda post koji je unet od strane drugog admina, pomenuti dugmići neće biti prikazani. Na slici 3.3.1.1. je prikazan primer posta sa odgovarajućim dugmićima za ažuriranje i brisanje.

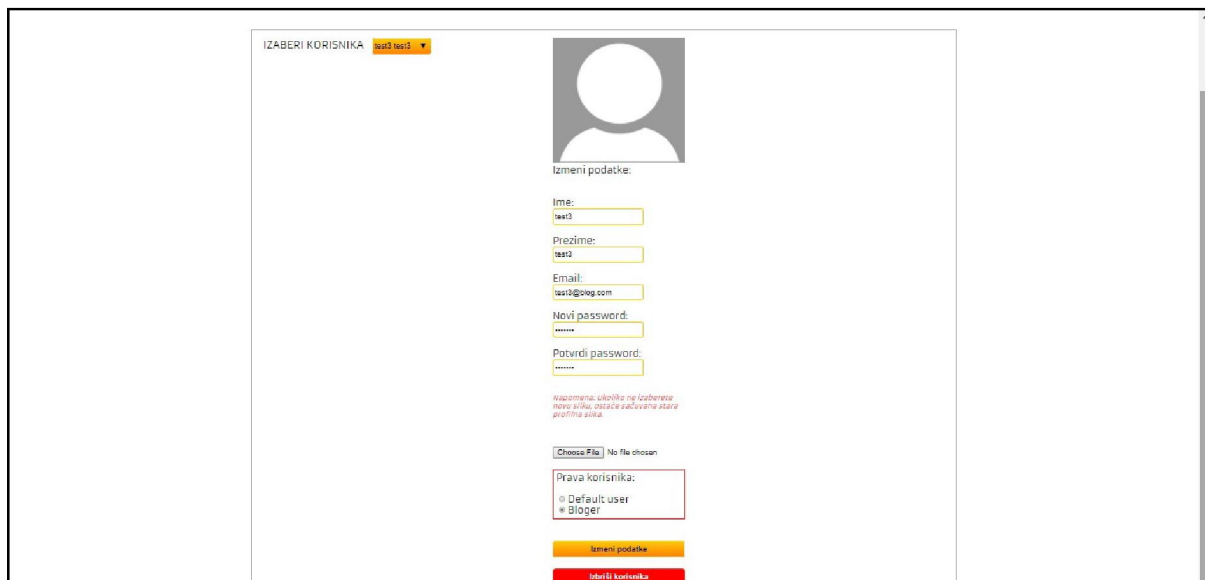


Slika 3.3.1.1. Izgled dugmića za ažuriranje posta od strane admina.

3.3.2. Upravljanje korisnicima

Pored upravljanja postovima, admin ima na raspolaganju još jednu važnu funkcionalnost. Izborom podopcije „Upravljaј korisnicima“, u okviru opcije „MANAGE“ u navigacionom meniju, otvara se stranica na kojoj admin ima mogućnost izmene podataka drugih korisnika. Slično kao i kod ažuriranja postova, admin na ovoj stranici ima mogućnost promene podataka za sve korisnike osim za druge admina.

Na vrhu stranice, u levom uglu, nalazi se padajuća lista iz koje je potrebno selektovati korisnika čiji se podaci menjaju. Nakon izbora korisnika na stranici, u odgovarajućim poljima učitavaju se korisnikova slika i podaci. Izgled formulara na pomenutoj stranici je prikazan na slici 3.3.2.1.



Slika 3.3.2.1. Izgled formulara za ažuriranje korisničkih podataka od strane admina.

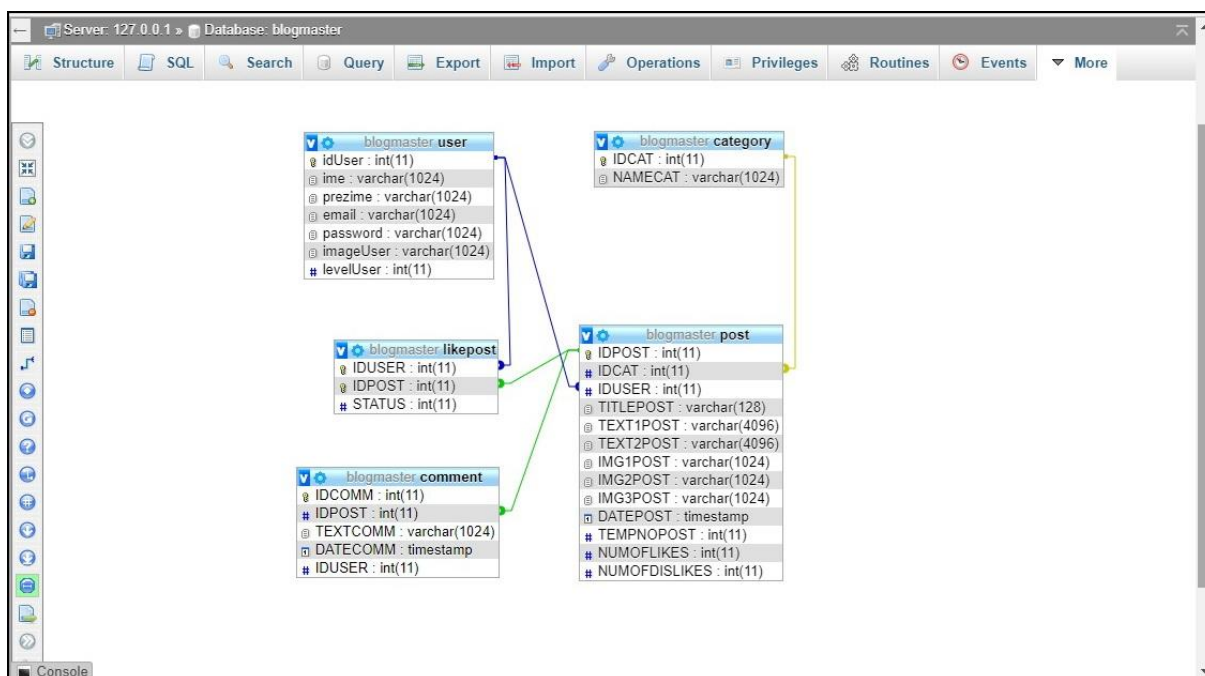
U dnu stranice, ispod dugmeta za aploudovanje slike, nalazi se opcija kojom admin ima mogućnost da promeni privilegije datog korisnika. Da bi izmene bile sačuvane, potrebno je potvrditi klikom na dugme „Izmeni podatke“. Pored ažuriranja korisničkih podataka, admin ima mogućnost brisanja korisnika. Nakon klika na dugme „Izbriši korisnika“, pojavljuje se prozor u kome se još jednom traži potvrda za brisanje. U slučaju brisanja korisnika, brišu se i svi njegovi postovi.

4. OPIS IMPLEMENTACIJE

U ovom poglavlju, pre svega, biće opisana struktura baze podataka. Nakon toga, biće predstavljen detaljan opis implementacije pojedinih funkcionalnosti, kao i pregled važnijih delova samog koda veb aplikacije.

4.1. Struktura baze podataka

Za potrebe veb aplikacije koristi se MySQL baza podataka. Za rad sa bazom podataka, u ovom slučaju, koristi se phpMyAdmin. Baza je nazvana „blogmaster“, a sastoji se od tabela „user“, „post“, „category“, „likepost“ i „comment“. Na slici 4.1.1. predstavljen je fizički model baze podataka sa odgovarajućim relacijama između tabela.



Slika 4.1.1. Stuktura baze podataka.

Tabela „user“ se sastoji od 7 kolona. Prva kolona, „idUser“, predstavlja primarni ključ. Ovo polje je jedinstveno i predstavlja identifikator svakog korisnika u tabeli. Nakon toga, slede kolone koje predstavljaju osnovne podatke korisnika tj. ime, prezime, email za logovanje i polje „password“ u kome se čuva kriptovana šifra korisnika. U polju „imageUser“ se čuva naziv slike za datog korisnika. Radi lakšeg manipulisanja, slike korisnika imaju naziv u formatu „user+idUser.jpg“ (npr. slika korisnika čije polje *idUser* ima vrednost 5, ima naziv „user5.jpg“). Ukoliko korisnik prilikom registracije ne aploaduje sliku, u polje za sliku datog korisnika biće upisan naziv „default.jpg“, kako bi se na stranicama koje zahtevaju sliku korisnika prikazala odgovarajuća slika koja je dodeljena korisnicima koji nisu uneli sliku. Poslednja kolona,

„levelUser“, definiše tip korisnika. Za obične korisnike, ova kolona ima vrednost 0, dok za blogere i admina ima vrednost 1 i 2, respektivno.

Tabela „category“ ima dve kolone, „IDCAT“ i „NAMECAT“. Polje „IDCAT“ je primarni ključ. Ova tabela se koristi za mapiranje id-a i imena svake kategorije. Na ovaj način je olakšan rad u slučaju nekih izmena, kao što je npr. dodavanje nove kategorije.

U tabeli „post“ se čuva najveći deo podataka, pa samim tim ima najveći broj kolona. Prva kolona, „IDPOST“, predstavlja primarni ključ i ovo polje je jedinstveni identifikator za svaki post u tabeli. Sledi kolona „IDCAT“ koja predstavlja strani ključ. Na osnovu vrednosti iz ovog polja, u tabeli „category“, može se videti naziv kategorije kojoj dati post pripada. Sledeća kolona, „IDUSER“, takođe predstavlja strani ključ, pomoću koga se u tabeli „user“ mogu videti podaci za datog korisnika. Slede kolone u kojima se čuvaju podaci koji se prikazuju na samoj aplikaciji, a to su: „TITLEPOST“ za naslov, „TEXT1POST“ i „TEXT2POST“ za tekst, kao i „IMG1POST“, „IMG2POST“ i „IMG3POST“ za čuvanje naziva slika za dati post. U koloni „DATEPOST“ čuva se datum unosa posta. Kolona „TEMPNOPOST“ se koristi za tip templejta i može imati vrednost 1,2 ili 3, u zavisnosti od templejta koji se izabere prilikom unosa od strane korisnika. Kolone „NUMOFLIKES“ i „NUMOFDISLIKES“ se koriste za čuvanje broja lajkova odnosno dislajkova i vrednosti u ovim kolonama se inkrementiraju svaki put kada korisnik lajkuje/dislajkuje dati post.

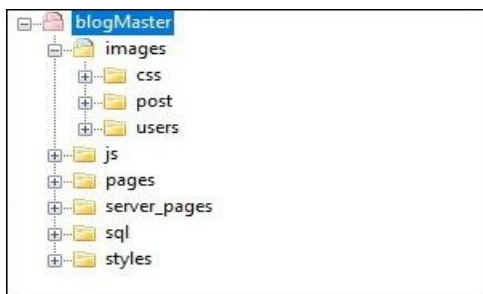
Tabela „likepost“ se sastoji od tri kolone: „IDUSER“, „IDPOST“ i „STATUS“. U prve dve kolone se čuva id korisnika koji je lajkovao/dislajkovao post, kao i id samog posta nad kojim je učinjena prethodna akcija. Informacija o samoj akciji korisnika čuva se u koloni „STATUS“ – vrednost 1 za lajk odnosno -1 za dislajk. Važno je napomenuti da u ovoj tabeli primarni ključ formiraju dva strana ključa „IDUSER“ i „IDPOST“. Kako primarni ključ u svakoj tabeli mora biti jedinstven, ovim je postignuto da svaki korisnik može samo jedanput da izvrši akciju koja se čuva u ovoj tabeli. Drugim rečima, korisnik je onemogućen da više puta lajkuje/dislajkuje isti post.

Na kraju, tabela „comment“ se koristi za čuvanje podataka o komentarima. Kolona „IDCOMM“ je primarni ključ i ona je jedinstvena za svaki komentar. Sledi kolona „IDPOST“ u kojoj se čuva id posta koji je komentarisano. Kolone „TEXTCOMM“ i „DATECOMM“ se koriste za čuvanje teksta samog komentara, kao i datuma njegovog unosa. Poslednja kolona, „IDUSER“, nosi informaciju o korisniku koji je uneo dati komentar.

4.2. Najvažniji delovi koda

4.2.1. Struktura projekta

Radi lakšeg praćenja opisa koda u daljem tekstu, na slici 4.2.1.1. je dat prikaz strukture projekta po folderima.



Slika 4.2.1.1. Struktura projekta po folderima.

U folderu „images“ nalaze se tri podfoldera u kojima se čuvaju odgovarajuće slike. U folderu „css“ se čuvaju slike koje se koriste u okviru fajla theme.css koji se nalazi u okviru foldera „styles“.

U folderima „users“ i „post“ se nalaze aploudovane slike korisnika odnosno slike koje su potrebne za prikaz odgovarajućeg posta. Sledi folder „js“ u okviru koga se nalaze JavaScript fajlovi neophodni za funkcionisanje aplikacije.

U folderu „pages“ se nalaze stranice koje se prikazuju u aplikaciji. U okviru ovih stranica generisani su pozivi ka serverskim stranicama koje se nalaze u okviru foldera „server_pages“. Ka stranicama iz foldera „server_pages“ se generišu AJAX zahtevi, dok one vraćaju odgovarajući odgovor ka klijentu.

Baza podataka, sa odgovarajućim podacima, smeštena je u okviru foldera „sql“. Prilikom učitavanja projekta, potrebno je importovati fajl .sql koji se nalazi u ovom folderu.

Na kraju, u folderu „styles“ se nalaze .css fajlovi koji su korišćeni u okviru aplikacije.

4.2.2. *Povezivanje sa bazom podataka i generisanje odgovora za klijenta*

Kako je na skoro svakoj serverskoj stranici potrebna komunikacija sa bazom podataka, pre generisanja odgovarajućih upita, potrebno je ostvariti konekciju sa bazom. Takođe, nakon završetka komunikacije, potrebno je zatvoriti konekciju ka bazi. Da bi se sprečilo „gomilanje“ koda, a i radi lakše manipulacije sa bazom, generisan je config.php koji se nalazi u okviru foldera „server_pages“. Ovaj fajl je učitavan u okviru svih stranica koje komuniciraju sa bazom. Sadržaj fajla je prikazan ispod.

```
<?php
function dbConnect() {
    $servername = "localhost";
    $username = "root";
    $password = "";
    $database = "blogmaster";

    // Kreira se konekcija
    $conn = new mysqli($servername, $username, $password, $database);
    $conn->set_charset("utf8");

    // Provera konekcije
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }
    return $conn;
}

function dbDisconnect($conn) {
    $conn->close();
}

function respondJson($status, $message) {
    $obj = new stdClass();
    $obj->status = $status;
    $obj->message = $message;
    die(json_encode($obj));
}??>
```

Funkcija *dbConnect* koristi se povezivanje sa korišćenom bazom. U okviru funkcije se nalaze podaci koji se koriste za konekciju, kao što su server na kome se baza nalazi, kredencijali za pristup, kao i ime baze. Takođe, podešena je i opcija za razmenu „UTF-8“ karaktera. Funkcija *dbDisconnect* zatvara predatu konekciju.

Pored navedenih, u ovom fajlu se nalazi i funkcija *respondJson* koju koriste serverske stranice kao odgovor na AJAX zahtev. Suština ove funkcije je da se nakon izvršene obrade na serverskoj strani, formira JSON (*JavaScript Object Notation*) objekat sa odgovarajućim statusom i porukom, koji se šalje klijentu. Na osnovu ovog odgovora, klijent može vršiti dalju obradu i može korisniku prikazati poruku u vezi uspešnosti izvršene akcije.

Glavna prednost ovog fajla jeste u tome što se svi podaci o bazi nalaze na jednom mestu. Na primer, ukoliko je promenjena lokacija baze na serveru, ili ako je promenjeno ime baze dovoljno je izmeniti config.php fajl.

4.2.3. Realizacija pretrage

U trenutku kada korisnik potvrdi pretragu, prvo se vrši validacija na klijentskoj strani. Nakon toga, bilo da je u pitanju osnovna ili napredna pretraga, on biva preusmeren na stranicu *searchResultsPage.php* koja se nalazi u okviru foldera „pages“. Pomenuta stranica, pored HTML koda kojim je definisan sam izgled stranice, sadrži i PHP logiku koja je neophodna da bi se na serversku stranicu koja je zadužena za obradu pretrage poslali odgovarajući parametri. U daljem tekstu biće objašnjen tok pretrage.

U slučaju da je korisnik kliknuo na dugme za potvrdu pretrage na formularu za osnovnu pretragu, prvo se izvršava funkcija *validateSearchForm* koja je smeštena u fajlu *script.js* u okviru foldera „js“. Uloga funkcije je da onemogući korisnika da pokuša pretragu, ukoliko nije unet pojam za pretragu. Sledi izgled same funkcije.

```
function validateSearchForm() {
    var text = $.trim($("#searchBar").val());
    if(text == "") {
        alert("Morate popuniti polje ! Unesite željeni pojam za pretragu.");
        return false;
    }
}
```

U slučaju da je korisnik izabrao naprednu pretragu, nakon klika na dugme za potvrdu, izvršava se funkcija *validateAdvancedSearchForm*. Kako se pretraga ne vrši samo po unetom pojmu, nije neophodno da korisnik unese pojam za pretragu. Na formularu za pretragu, u padajućem meniju izlistani su svi autori koji trenutno imaju makar jedan post, a na prvom mestu ponuđena je opcija svi autori. Ni u ovom slučaju nije potrebno ograničavati korisnika, jer u slučaju da korisnik ne izabere neku od opcija iz padajućeg menija, ostaće izabrana opcija kojom se vrši pretraga po svim postojećim autorima. Slično, kada korisnik ne izabere ni jednu kategoriju, pretraga će se vršiti po svim postojećim kategorijama. Gore navedena funkcija vrši validaciju unetog opsega datuma za pretragu. Ako korisnik ne unese početni datum za pretragu, postavlja se vrednost „2018-01-01“, a u slučaju da nije unet krajnji datum, uzima se trenutni datum. Implementacija funkcije prikazana je u nastavku.

```

function validateAdvancedSearchForm() {

    var dateFrom = $("#dateFrom").val();
    var dateTo = $("#dateTo").val();
    if(dateFrom == ""){
        $("#dateFrom").val("2018-01-01");
    }

    if(dateTo == ""){
        var date = getCurrentDate();
        $("#dateTo").val(date);
    }
}

```

Kao što je gore pomenuto, nakon validacije, sledi obrada na stranici searchResultsPage.php. Logika implementirana na pomenutoj stranici je prikazana ispod, nakon čega sledi detaljno objašnjenje svake celine.

```

<?php
if(isset($_POST['typeOfSearch'])){
    if(isset($_POST['textForSearch'])){
        $textForSearch = trim($_POST['textForSearch']);
    }else{
        $textForSearch='';
    }

    $typeOfSearch = $_POST['typeOfSearch'];

    if($typeOfSearch == "regular"){
        echo <script>
        loadPostsOnSearchResultPage('".$textForSearch."',
        "."".$typeOfSearch."');</script>";
    }else{
        //advanced search
        if(isset($_POST['dateFrom']) && isset($_POST['dateTo']) &&
        isset($_POST['idSelectedUser'])){
            $idSelectedUser = intval($_POST['idSelectedUser']);
            $kat = array(0,0,0,0,0);
            if(isset($_POST['kategorije'])){
                $kategorije = $_POST['kategorije'];
                if(!empty($kategorije)){
                    for($i=0; $i < 5; $i++){
                        if(!empty($kategorije[$i])){
                            $kat[$i] = $kategorije[$i];
                        }else{
                            $kat[$i] = 0;
                        }
                    }
                }
            }else{
                //pretraga po svim kategorijama
                $kat = array(1,2,3,4,5);
            }
            $dateFrom = $_POST['dateFrom'];
            $dateTo = $_POST['dateTo'];

```

```

        echo "<script>
        loadPostsOnSearchResultPageAdvanced
        ('".$textForSearch."', ".$typeOfSearch."', "
        ".$kat[0]."', ".$kat[1]."', ".$kat[2]."', "
        ".$kat[3]."', ".$kat[4]."', ".$dateFrom."', "
        ".$dateTo."', ".$idSelectedUser.'");
        </script>";
    }else{
        echo "<div id='noResult'>Došlo je do greške advanced
        search, pokušajte opet!</div>";
    }
}
}
}
}
}
?>

```

Pre svega, na osnovu promenljive *\$typeOfSearch* pravi se grananje za osnovnu i naprednu pretragu. U slučaju osnovne pretrage poziva se funkcija *loadPostsOnSearchResultPage* kojoj se prosleđuju parametri pojam i tip pretrage. Pomenuta funkcija je definisana u *script.js* fajlu i njen zadatak je da generiše AJAX zahtev sa prosleđenim parametrima ka stranici koja generiše rezultat pretrage. Ona se zove *loadPostsOnSearchResultPage.php* i smeštena je u okviru foldera „*server_pages*“. U okviru grane za naprednu pretragu, u pomoćni niz *\$kat* se smeštaju vrednosti id-jeva izabranih kategorija (naziv svake kategorije je mapiran sa odgovarajućim id-jem u tabeli „*category*“), dok se za neizabrane kategorije smešta vrednost 0. U slučaju da korisnik nije izabrao nijednu ponudenu kategoriju, u pomenuti niz se smeštaju id-jevi svih kategorija (u ovom slučaju, vrednosti od 1 do 5). Zatim, analogno slučaju za osnovnu pretragu, poziva se odgovarajuća funkcija *loadPostsOnSearchResultPageAdvanced* kojoj se proseđuju svi potrebni parametri.

Slično osnovnoj, za naprednu pretragu se koristi funkcija *loadPostsOnSearchResultPageAdvanced*. Ona služi za slanje AJAX zahteva ka serverskoj stranici *loadPostsOnSearchResultPage.php* koja generiše rezultat pretrage. Navedene funkcije prihvataju parametre pretrage i prosleđuju ih serverskoj stranici. Rezultat pretrage se prikazuje na stranici *searchResultsPage.php* u okviru `<div id='MainSearchResultPost'>` taga. Implementacija pomenutih funkcija prikazana je u nastavku.

```

function loadPostsOnSearchResultPage(textForSearch, typeOfSearch){
    var textForSearch = encodeURIComponent(textForSearch);

    $("#MainSearchResultPosts").load("../server_pages/loadPostsOnSearchResultPage.php?textForSearch="+textForSearch+"&typeOfSearch="+typeOfSearch);
}

function loadPostsOnSearchResultPageAdvanced(textForSearch, typeOfSearch, kat1, kat2, kat3, kat4, kat5, dateFrom, dateTo, idSelectedUser){
    var textForSearch = encodeURIComponent(textForSearch);

    $("#MainSearchResultPosts").load("../server_pages/loadPostsOnSearchResultPage.php?textForSearch="+textForSearch+"&typeOfSearch="+typeOfSearch+"&kat1="+kat1+"&kat2="+kat2+"&kat3="+kat3+"&kat4="+kat4+"&kat5="+kat5+"&dateFrom="+dateFrom+"&dateTo="+dateTo+"&idSelectedUser="+idSelectedUser);
}

```

Sledi opis stranice `loadPostsOnSearchResultPage.php` koja je u stvari srž pretrage. Na njoj se generišu svi upiti na osnovu primljenih parametara. Upiti se izvršavaju nad odgovarajućim tabelama i rezultat vraćaju nazad na klijentsku stranu kao odgovor AJAX zahtevu.

Prvo se vrši provera da li je prosleđen parametar za tip pretrage i njegova vrednost se smešta u promenljivu `$typeOfSearch`. Nakon toga, se postavlja vrednost promenljive `$textForSearch` u kojoj se čuva pojam pretrage koji je uneo korisnik.

```
<?php
include("../server_pages/config.php");
if(isset($_GET['typeOfSearch'])){
    if(isset($_GET['textForSearch'])){
        $textForSearch = urldecode($_GET['textForSearch']);
    }else{
        $textForSearch='';
    }
    $typeOfSearch = $_GET['typeOfSearch'];
}
```

Za potrebe paginacije tj. prikaza rezultata pretrage po stranicama, neophodno je setovanje promenljive `$postsPage` u kojoj se čuva redni broj stranice na kojoj se prikazuje rezultat. U promenljivoj `$numberOfPostsIndex` se čuva broj postova koji se prikazuje na svakoj stranici. U ovom slučaju, to je konstantna vrednost i iznosi 5 postova po stranici. Takođe, svaki put kada se poziva upit, neophodan je i parametar čija se vrednost računa i smešta u promenljivu `$offset`.

```
if(isset($_GET['postsPage'])){
    $postsPage = intval($_GET['postsPage']);
}else{
    $postsPage = 1;
}
$numberOfPostsIndex = 5;
$offset = ($postsPage-1)*$numberOfPostsIndex;
```

Sledi grananje na osnovnu i naprednu pretragu i poziv odgovarajućih funkcija kojima se prosleđuju parametri za generisanje upita. U promenljivoj `$brojRezultata` se smešta broj nađenih postova, dok se u promenljivoj `$result` čuva niz nađenih vrsta i ona predstavlja rezultat pretrage.

```
switch($typeOfSearch){
    case "regular":{
        $brojRezultata = getNumOfResRegSearch($textForSearch);
        $result = regularSearch($textForSearch, $offset,
            $numberOfPostsIndex);
        break;
    }

    case "advanced":{
        if(isset($_GET['idSelectedUser'])){
            $idSelectedUser = intval(urldecode($_GET['idSelectedUser']));
        }
        $textForSearch = urldecode($_GET['textForSearch']);
    }
}
```

```

$kat1 = intval($_GET['kat1']);
$kat2 = intval($_GET['kat2']);
$kat3 = intval($_GET['kat3']);
$kat4 = intval($_GET['kat4']);
$kat5 = intval($_GET['kat5']);

$dateFrom = $_GET['dateFrom'];
$dateTo = $_GET['dateTo'];

//provera da li je selektovan konkretni autor
if($idSelectedUser == -1){

    $brojRezultata = getNumOfResRegAdvancedSearch($textForSearch,
    $kat1, $kat2, $kat3, $kat4, $kat5, $dateFrom, $dateTo);

    $result = advancedSearch($textForSearch, $kat1, $kat2, $kat3,
    $kat4, $kat5, $dateFrom, $dateTo, $offset,
    $numberOfPostsIndex);
} else{
    //trazi po autoru
    $brojRezultata =
    getNumOfResRegAdvancedSearchByAuthor($textForSearch, $kat1,
    $kat2, $kat3, $kat4, $kat5, $dateFrom, $dateTo,
    $idSelectedUser);

    $result = advancedSearchByAuthor($textForSearch, $kat1, $kat2,
    $kat3, $kat4, $kat5, $dateFrom, $dateTo, $offset,
    $numberOfPostsIndex, $idSelectedUser);
}
break;
}
}

```

U slučaju osnovne pretrage pozivaju se funkcije *getNumOfResRegSearch* i *regularSearch* kojima se prosleđuju neophodni parametri. Njihova implementacija je prikazana u tekstu ispod.

```

function getNumOfResRegSearch($textForSearch){

    $conn = dbConnect();

    $query = "SELECT count(IDPOST) as brojRez FROM post
    WHERE TITLEPOST LIKE ?
    OR TEXT1POST LIKE ?
    OR TEXT2POST LIKE ?";

    $searchBy = "%".$textForSearch."%";
    $stmt = $conn->prepare($query);
    $stmt->bind_param('sss', $searchBy, $searchBy, $searchBy);
    $stmt->execute();

    $result = $stmt->get_result();
    $row = $result->fetch_assoc();
    $brojRezultata = $row['brojRez'];
}

```

```

        dbDisconnect($conn);
        return $brojRezultata;
    }

    function advancedSearch($textForSearch, $kat1, $kat2, $kat3, $kat4, $kat5,
        $dateFrom, $dateTo, $offset, $numberOfPostsIndex){

        $conn = dbConnect();

        $query = "SELECT * FROM post
            WHERE
                (TITLEPOST LIKE ? OR TEXT1POST LIKE ? OR TEXT2POST LIKE ?)
                AND IDCAT in(?,?,?,?,?)
                AND (DATEPOST BETWEEN ? AND ? )
                LIMIT ?, ?";

        $searchBy = "%".$textForSearch."%";

        $stmt = $conn->prepare($query);
        $stmt->bind_param('sssiiiiissii', $searchBy, $searchBy, $searchBy, $kat1,
            $kat2, $kat3, $kat4, $kat5, $dateFrom, $dateTo, $offset,
            $numberOfPostsIndex);
        $stmt->execute();

        $result = $stmt->get_result();

        dbDisconnect($conn);

        return $result;
    }

```

Slično osnovnoj, za naprednu pretragu se pozivaju funkcije *getNumOfResRegAdvancedSearchByAuthor* i *advancedSearchByAuthor*.

```

    function getNumOfResRegAdvancedSearchByAuthor($textForSearch, $kat1, $kat2,
        $kat3, $kat4, $kat5, $dateFrom, $dateTo, $idSelectedUser){

        $conn = dbConnect();

        $query = "SELECT count(IDPOST) as brojRez FROM post
            WHERE
                (TITLEPOST LIKE ? OR TEXT1POST LIKE ? OR TEXT2POST LIKE ?)
                AND IDCAT in(?,?,?,?,?)
                AND IDUSER = ?
                AND (DATEPOST BETWEEN ? AND ? )";

        $searchBy = "%".$textForSearch."%";

        $stmt = $conn->prepare($query);
        $stmt->bind_param('sssiiiiiss',
            $searchBy, $searchBy, $searchBy, $kat1, $kat2, $kat3, $kat4, $kat5,
            $idSelectedUser, $dateFrom, $dateTo);
        $stmt->execute();

        $result = $stmt->get_result();

        $row = $result->fetch_assoc();
    }

```



```

$brojRezultata = $row['brojRez'];

dbDisconnect($conn);

return $brojRezultata;
}

function advancedSearchByAuthor($textForSearch, $kat1, $kat2, $kat3, $kat4,
$kat5, $dateFrom, $dateTo, $offset, $numberOfPostsIndex, $idSelectedUser){

    $conn = dbConnect();

    $query = "SELECT * FROM post
            WHERE
            (TITLEPOST LIKE ? OR TEXT1POST LIKE ? OR TEXT2POST LIKE ?)
            AND IDCAT in(?,?,?,?,?)
            AND IDUSER = ?
            AND (DATEPOST BETWEEN ? AND ? )
            LIMIT ?, ?";

    $searchBy = "%".$textForSearch."%";

    $stmt = $conn->prepare($query);
    $stmt->bind_param('ssiiiiissii',
    $searchBy, $searchBy, $searchBy, $kat1, $kat2, $kat3, $kat4, $kat5,
    $idSelectedUser, $dateFrom, $dateTo, $offset, $numberOfPostsIndex);
    $stmt->execute();

    $result = $stmt->get_result();

    dbDisconnect($conn);

    return $result;
}

```

Nakon izvršenih upita, vrši se provera o broju nađenih rezultata na osnovu čega se prikazuje rezultat pretrage ili odgovarajuća poruka da ne postoji rezultat za zadate parametre. Takođe, za potrebe paginacije, u odgovarajuća skrivena polja smeštaju se parametri pretrage.

```

if($result->num_rows > 0){

    if($typeOfSearch == "advanced"){
        echo"
        <input type='hidden' id='kat1' value='$kat1' />
        <input type='hidden' id='kat2' value='$kat2' />
        <input type='hidden' id='kat3' value='$kat3' />
        <input type='hidden' id='kat4' value='$kat4' />
        <input type='hidden' id='kat5' value='$kat5' />
        <input type='hidden' id='dateFromhidden' value='$dateFrom' />
        <input type='hidden' id='dateTohidden' value='$dateTo' />

```

```

<input type='hidden' id='idSelectedUser' value='$idSelectedUser' />";
}

echo "
<input type='hidden' id='typeOfSearch' value='$typeOfSearch' />
<input type='hidden' id='textForSearch' value='$textForSearch' />
<div id='NumOfSearchResults'> Rezultati pretrage za pojam:
\"$textForSearch\"
</div>Pronađeno ukupno: $brojRezultata rezultata</div></div><br/>";

while ($row = $result->fetch_assoc()) {
    $text1Post = htmlentities(substr($row['TEXT1POST'],0,300));
    echo "<div class='authorPost'><a href='postPage.php?postId="
    .$row['IDPOST']. "'><h3>".htmlentities($row['TITLEPOST'])
    ."</h3></a><p class='datePost'> Datum objave:"
    .$row['DATEPOST']. "</p>
    <a href='postPage.php?postId=".$row['IDPOST']. "'>
    <div class='authorImg'>
    <img src='../images/post/slikalpost".$row['IDPOST']. ".jpg'
    onerror=\"this.src = '../images/post/error.jpg';\">
    </div>
    </a>
    <div>\"
    .$text1Post. "</div></div><hr>";
}

} else {
    echo "<div id='noResult' >Nema rezultata pretrage!</div>";
}
}

```

Na samom kraju, računa se broj strana za prikaz rezultata i prikazuje se paginacija.

```

echo "<center>Idi na stranu &nbsp;";
for($i=1; $i<=$brStrana; $i++){
    echo "<span data-idPage='$i' class='selectPageSearch
    selectPageNumber'>";
    if($i == $postsPage) echo "<sup> $i </sup>";
    else echo "$i";
    echo "</span>";
}
echo "</center>";
}
}

```

Klikom na neku od ponuđenih stranica, ponovo se poziva stranica `loadPostsOnSearchResultPage.php` kojoj se prosleđuju parametri za pretragu izvučeni iz skrivenih polja. Ovo implementirano u *jQuery* funkciji koja je se nalazi u okviru fajla „script.js“ i prikazana je u nastavku.

```

$(document).on("click", ".selectPageSearch", function() {

    var elem = this;
    var typeOfSearch = $("#typeOfSearch").val();
    var textForSearch = encodeURIComponent($("#textForSearch").val());

```

```

if(typeof $("#kat1").val() !== "undefined"){
    var kat1 = $("#kat1").val();
}

if(typeof $("#kat2").val() !== "undefined"){
    var kat2 = $("#kat2").val();
}

if(typeof $("#kat3").val() !== "undefined"){
    var kat3 = $("#kat3").val();
}

if(typeof $("#kat4").val() !== "undefined"){
    var kat4 = $("#kat4").val();
}

if(typeof $("#kat5").val() !== "undefined"){
    var kat5 = $("#kat5").val();
}

if($("#dateFromhidden").val() !== ""){
    var dateFrom = $("#dateFromhidden").val();
}

if(typeof $("#dateTohidden").val() !== "undefined"){
    var dateTo = $("#dateTohidden").val();
}

var idSelectedUser = encodeURI($("#idSelectedUser").val());

var brStrane = $(elem).attr('data-idPage');

$("#MainSearchResultPosts").load("../server_pages/loadPostsOnSearchResultPage.php?textForSearch="+textForSearch+"&typeOfSearch="+typeOfSearch+"&postPage="+brStrane+"&kat1="+kat1+"&kat2="+kat2+"&kat3="+kat3+"&kat4="+kat4+"&kat5="+kat5+"&dateFrom="+dateFrom+"&dateTo="+dateTo+"&idSelectedUser="+idSelectedUser);
});

```

4.2.4. Realizacija dodavanja posta

Nakon što korisnik potvrdi unošenje novog posta u formularu izabranog templejta, uneti podaci se POST metodom šalju na stranicu `addPost.php` koja se nalazi u okviru foldera „`server_pages`“. Na samom početku vrši se povezivanje sa bazom i smeštanje primljenih podataka u odgovarajuće promenljive, koje će kasnije biti upotrebljene prilikom formiranja upita. U promenljivu `$templateNo` se smešta redni broj templejta, u `$category` kategorija kojoj post pripada, dok se u `$title` smešta naslov posta. Kako bi bio poznat i autor posta, u promenljivu `$idUser` se smešta id ulogovanog korisnika koji se uzima iz kukija.

```

<?php
include("config.php");
$conn = dbConnect();
$templateNo = intval($_POST['tempNo']);
$idUser = intval($_COOKIE['loggedUser']['idUser']);

```

```
$category = $_POST['category'];
$title = $_POST['title'];
```

Kako svaki od tri ponuđena templejta ima makar jedno polje za prikaz unetog teksta, u promenljivu *\$text1* se učitava tekst koji je korisnik uneo prvo polje. U slučaju drugog i trećeg templejta, ponuđena su dva polja za unos teksta. Sledi ispitivanje koji templejt je izabrao korisnik i u slučaju da to nije prvi, u promenljivu *\$text2* se smešta tekst koji je korisnik uneo u drugo polje.

```
$text1 = $_POST['text1Temp'];
if($templateNo > 1){
    $text2 = $_POST['text2Temp'];
}else{
    $text2 = 0;
}
```

Pošto kolona „IDPOST“ u tabeli „post“ nije tipa AUTO_INCREMENT, jer je strani ključ u drugim tabelama, prilikom unosa svakog posta potrebno je odrediti id pod kojim će se uneti u tabelu.

```
$query = "SELECT MAX(IDPOST)+1 as nextId FROM post";
$stmt = $conn->prepare($query);
$stmt->execute();
$result = $stmt->get_result();
$row = $result->fetch_assoc();
$nextId = $row['nextId'];
```

Slično kao i za polja za unos teksta, u zavisnosti od templejta varira broj slika koje korisnik može da unese. U svakom slučaju potrebno je učitati makar dve slike, a dodatno ako je u pitanju prvi templejt, učitava se i treća slika. U slučaju da korisnik nije uneo sliku, ili da slika nije uspešno aploudovana, pamti se pod imenom "default.jpg". Uspešno aploudovane slike se čuvaju u podfolderu „post“, u okviru foldera „images“. U nastavku je prikazan kod kojim je ovo realizovano.

```
if(is_array($_FILES)){

    if(is_uploaded_file($_FILES['slika1Temp']['tmp_name'])){
        $sourcePath = $_FILES['slika1Temp']['tmp_name'];
        $targetPath = "../images/post/slikalpost".$nextId.".jpg";
        if(move_uploaded_file($sourcePath,$targetPath)){
            $slika1 = "slikalpost".$nextId.".jpg";
        }else{
            $slika1 = "default.jpg"; //kada slika nije aploudovana
        }
    }else{
        $slika1 = "default.jpg"; //kada korisnik nije uneo sliku
    }
}

if(is_array($_FILES)){

    if(is_uploaded_file($_FILES['slika2Temp']['tmp_name'])){
```

```

        $sourcePath = $_FILES['slika2Temp']['tmp_name'];
        $targetPath = "../images/post/slika2post".$nextId.".jpg";
        if(move_uploaded_file($sourcePath,$targetPath)){
            $slika2 = "slika2post".$nextId.".jpg";
        }else{
            $slika2 = "default.jpg";
        }
    }else{
        $slika2 = "default.jpg";
    }
}

if($templateNo == 1){
    if(is_array($_FILES)){
        if(is_uploaded_file($_FILES['slika3Temp']['tmp_name'])){
            $sourcePath = $_FILES['slika3Temp']['tmp_name'];
            $targetPath = "../images/post/slika3post".$nextId.".jpg";
            if(move_uploaded_file($sourcePath,$targetPath)){
                $slika3 = "slika3post".$nextId.".jpg";
            }else{
                $slika3 = "default.jpg";
            }
        }else{
            $slika3 = "default.jpg";
        }
    }
}
}
}

```

Nakon što su slike smeštene u odgovarajući folder na serveru, sledi dodavanje novog posta u tabelu. U zavisnosti od templejta, izvršava se jedan od tri slučaja navedenih u nastavku.

```

switch($templateNo){
    case 1: {
        $date = date("Y-m-d H:i:s");
        $query = "INSERT INTO post(IDPOST, IDCAT, TITLEPOST,
            TEXT1POST, DATEPOST, TEMPNOPOST, IDUSER, IMG1POST, IMG2POST,
            IMG3POST) VALUES(?,?,?,?,?,?,?,?,?,?)";
        $stmt = $conn->prepare($query);
        $stmt->bind_param('iisssiisss', $nextId, $category, $title
            , $text1, $date, $templateNo, $idUser,
            $slika1, $slika2, $slika3);
        if(!$stmt->execute()){
            respondJson("error", "error in query: " . $conn->error);
        }

        dbDisconnect($conn);
        respondJson("Success", $nextId);
        break;
    }

    case 2: {
        $date = date("Y-m-d H:i:s");
        $query = "INSERT INTO post(IDPOST, IDCAT, TITLEPOST,
            TEXT1POST, TEXT2POST, DATEPOST, TEMPNOPOST, IDUSER, IMG1POST,
            IMG2POST) VALUES(?,?,?,?,?,?,?,?,?,?)";
        $stmt = $conn->prepare($query);
    }
}

```

```

$stmt->bind_param('iissssiiss', $nextId , $category, $title
, $text1, $text2, $date, $templateNo, $idUser
, $slika1, $slika2);
if(!$stmt->execute()){
    respondJson("error", "error in query: " . $conn->error);
}

dbDisconnect($conn);
respondJson("Success", $nextId);
break;
}

case 3: {
    $date = date("Y-m-d H:i:s");
    $query = "INSERT INTO post(IDPOST, IDCAT, TITLEPOST,
    TEXT1POST, TEXT2POST, DATEPOST, TEMPNOPOST, IDUSER, IMG1POST,
    IMG2POST) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
    $stmt = $conn->prepare($query);
    $stmt->bind_param('iissssiiss', $nextId , $category, $title
, $text1, $text2, $date, $templateNo, $idUser
, $slika1, $slika2);
    if(!$stmt->execute()){
        respondJson("error", "error in query: " . $conn->error);
    }

    dbDisconnect($conn);
    respondJson("Success", $nextId);
    break;
}
}
?>

```

Po izvršenju svakog upita, pomoću funkcije *respondJson*, klijentu se vraća odgovor sa statusom, kao i vrednost id-ja pod kojim je post unesen u tabelu.

4.2.5. Prikaz posta

Za prikazivanje posta zadužena je stranica *postPage.php*, smeštena u okviru foldera „server_pages“. Na njoj je implementirana sva potrebna logika kojom se određuje šta će biti prikazano na stranici. Izgled stranice varira u zavisnosti od templejta posta, kao i da li ulogovani korisnik ima admin prava. Sledi detaljan opis stranice.

Na samom vrhu stranice, učitani su neophodni fajlovi u okviru kojih se nalazi HTML kod koji je zajednički za svaku stranu.

```

<?php include("head.php");?>
<div id="header">
    <?php include("header.php");?>
</div><br/><br/>
<div id="wrapper">
    <?php include("navigationCategories.php");?>
    <div id="main">
        <div id="leftMainPost">
            <?php include('modalWindows.php');?>

```

Parametar koji je neophodan za pronalazak posta u tabeli „post“ jeste njegov id. On se stranici postPage.php prosleđuje GET metodom. Sledi povezivanje na bazu i ispitivanje da li postoji post čiji id je poslat.

```
$postId = intval($_GET['postId']);
include("../server_pages/config.php");
$conn = dbConnect();
$query = "SELECT * FROM post
        JOIN user ON post.IDUSER = user.idUser
        WHERE IDPOST = $postId";
$stmt = $conn->prepare($query);
if(!$stmt->execute()){
    respondJson("error", "error in query: " . $conn->error);
}
$result = $stmt->get_result();
```

Ukoliko nije nađen post u tabeli, prikazuje se odgovarajuća poruka i prekida se izvršavanje koda. U suprotnom, u odgovarajuće promenljive se smeštaju podaci iz tabele.

```
if($result->num_rows == 0){
    die("Traženi post ne postoji!");
}else{

    $row = $result->fetch_assoc();
    $postTemplate = intval($row['TEMPNOPOST']);
    $postCategory = $row['IDCAT'];
    $brojLajkova = $row['NUMOFLIKES'];
    $brojDislajkova = $row['NUMOFDISLIKES'];
    $postsAuthorLevel = $row['levelUser'];
    $postsAuthorId = $row['idUser'];
```

Zatim se vrši provera koji korisnik je trenutno ulogovan i da li on ima admin prava. U slučaju da je ulogovani korisnik admin, na vrhu stranice je potrebno prikazati dugmiće za brisanje i ažuriranje datog posta. Međutim, potrebno je izvršiti i proveru da li je autor posta neki drugi admin. U tom slučaju, kako admin nema prava da briše i ažurira postove drugih admina, ove dugmiće ne treba prikativati. Logika kojom je ovo realizovano sledi u nastavku.

```
if(isset($_COOKIE['loggedUser']['levelUser']) &&
isset($_COOKIE['loggedUser']['idUser'])){
    $levelUser = $_COOKIE['loggedUser']['levelUser'];
    $idUserLoggedUser = $_COOKIE['loggedUser']['idUser'];
    if($levelUser == 2){ //provera da li je ulogovani korisnik admin

        if($postsAuthorLevel == 2){ //provera da li je autor posta admin

            if($postsAuthorId == $idUserLoggedUser){
                //prikazuju se dugmici samo ako post pripada ulogovanom adminu
                echo"<div id='authorButtonsManagePost'>
                <a href='../pages/azurirajPost.php?idPost=$postId'>
                <input type='submit' class='headerButton' value='Ažuriraj'>
                </a> &nbsp;";
```

```

        echo"<form class='obrisiPostAdmin'>
        <input hidden name='idPost' value='\".$postId.\"'>
        <input hidden name='tempNoPost' value='\".$postTemplate.\"'>
        <input type='text' name='action' value='delete' hidden />
        <input type='submit' class='headerButton' value='Obriši'>
        </form></div>";

    }

    }else{//ako autor posta nije admin svakako se prikazuju dugmici
        echo"<div id='authorButtonsManagePost'>
        <a href='../pages/azurirajPost.php?idPost=$postId'>
        <input type='submit' class='headerButton' value='Ažuriraj'>
        </a> &nbsp;";

        echo"<form class='obrisiPostAdmin'>
        <input hidden name='idPost' value='\".$postId.\"'>
        <input hidden name='tempNoPost' value='\".$postTemplate.\"'>
        <input type='text' name='action' value='delete' hidden />
        <input type='submit' class='headerButton' value='Obriši'>
        </form></div>";

    }

}

}

}

```

Sledi AJAX zahtev za prikaz samog posta. U zavisnosti od vrednosti promenljive *\$postTemplate* poziva se jedna od tri moguće stranice i prosleđuje joj se id posta koji je potrebno prikazati.

```

<div id="postView">

</div>

<?php echo ('<script>
$("#postView").load("post'.$postTemplate.'.template.php?idPost='.$postId.'");
</script>');
?>

```

Na primer, u slučaju da se prikazuje post čiji templejt ima vrednost 1, generiše se zahtev ka stranici *post1template.php* i prosleđuje joj se željeni id. Odgovarajuća stranica kao odgovor na AJAX zahtev vraća HTML kod koji se učitava u *<div id="postView">*. Sadržaj stranice *post1template.php* prikazan je ispod.

```

<?php
$postId = intval($_GET['idPost']);
include("../server_pages/config.php");
$conn = dbConnect();
$query = "SELECT * FROM post po
        JOIN user us on po.iduser = us.iduser
        WHERE po.IDPOST = $postId";
$stmt = $conn->prepare($query);
if(!$stmt->execute()){
    respondJson("error", "error in query: " . $conn->error);
}

```



```

    }
    $result = $stmt->get_result();
    $row = $result->fetch_assoc();

?>
    <p id="naslovPostaView"><?php echo htmlentities($row['TITLEPOST']); ?></p>
    <div id="autorDatum">
        <div id="autor">
            <label>Autor: <span id="autorSpan" class="gray"
onclick="goToAuthorPage(<?php echo $row['idUser']?>)"><?php
echo ($row['ime']); ?></span></label>
        </div>
        <div id="datumObjave">
            <label>Datum objave: <span class="gray"><?php echo $row['DATEPOST'];
?></span> </label>
        </div>
    </div>
    
    <script> runSlider(<?=$postId?>); </script>
    <div id="textTeplatel" class="ow">
        <?php echo htmlentities($row['TEXT1POST']); ?>
    </div>

```

Sledi povratak na postPage.php stranu. Po prikazanom postu, u desnom delu stranice u okviru dela „Povezani postovi“, potrebno je prikazati postove koji pripadaju istoj kategoriji kao tekući post. Ova lista se prikazuje u HTML tagu `<div id="rightMainPost">`, a kod kojim je ovo realizovano sledi u nastavku.

```

<div id="rightMainPost">
    <p id='povezaniPostoviNaslov'>Povezani postovi ...</p>
    <ul id='povezaniPostovi'>
        <?php
$query = "SELECT * FROM post WHERE IDCAT = $postCategory
        ORDER BY DATEPOST DESC
        LIMIT 30";
$stmt = $conn->prepare($query);
if(!$stmt->execute()){
    respondJson("error", "error in query: " . $conn->error);
}
$result = $stmt->get_result();
while($row = $result->fetch_assoc()){
    $idPost = $row['IDPOST'];
    $titlePost = htmlentities($row['TITLEPOST']);
    //Da se ne bi prikazao trenutni post u povezanim postovima
    if($idPost != $postId){
        echo "<a href='postPage.php?postId=$idPost'><li>$titlePost</li></a>";
    }
}
?>
    </ul>
</div>
</div>

```

U donjem delu stranice postPage.php, potrebno je još prikazati lajkove/dislajkove, komentare za dati post, kao i futer.

```
<div id="lajkoviKomentari" >
  <div>
    <i id='iconLike' class="fa fa-thumbs-o-up" data-idPost='<?php echo
    $postId; ?>'>
    </i>&nbsp;
    <span id='numberOfLikes'><?php echo $brojLajkova; ?></span> &nbsp;
    &nbsp;
    <i id='iconDislike' class="fa fa-thumbs-o-down" data-idPost='<?php
    echo $postId; ?>'>
    </i>&nbsp;
    <span id='numberOfDislikes'><?php echo $brojDislajkova; ?></span>
  </div>
  <br/>
  <i id='iconUnesiKom' class="fa fa-plus-square accordion"></i>
  &nbsp; Unesi komentar
  <div id="unos_komentara">
    <form id='unesiKomentar'>
      <input name='idPost' value='<?php echo $postId ?>' hidden>
      <textarea id="komentar" name="komentarTextArea"
        maxlength="200" rows="10" cols="60"
        placeholder="Ovde uneti komentar.
        Maksimalna dužina je 200 karaktera." required>
      </textarea><br/>
      <input type='submit' value='Pošalji' name='posaljiKom'
        class='headerButton' />
    </form>
  </div><br/><br/>
  <i id='iconPrikaziKom' class="fa fa-plus-square
  accordionPrikazKom"></i>&nbsp; Pogledaj sve komentare
  <div id="svi_komentari">
  </div>
  <script>
    loadAllPostComments (<?php echo $postId ?>);
  </script>
  <br/><br/>
  </div>
  <script>
    toggleUnesiKomentar ();
  </script>
  <script>
    togglePrikaziKomentare ();
  </script>
</div>
<?php include ("footer.php"); ?>
```

Funkcija *loadAllPostComments* se nalazi u fajlu "script.js" i ona generiše AJAX zahtev ka stranici *loadAllPostComments.php*. Na ovoj stranici se na osnovu prosleđene vrednosti id posta, vrši pretraga svih unetih komentara za dati post. Odgovor koji vraća ova stranica se prikazuje u HTML tagu `<div id="svi_komentari">`.

Funkcije *toggleUnesiKomentar* i *togglePrikaziKomentare* se takođe nalaze u fajlu "script.js". One se koriste radi boljeg korisničkog iskustva i prikaza određenih delova stranice posta. Klikom na simbol „+“, korisnik ima mogućnost da prikaže/sakrije deo stranice za unos komentara, kao i deo u okviru koga su učitani svi komentari za dati post.

U okviru ovog poglavlja, prikazana je struktura baze podataka korišćene u aplikaciji. Objašnjena je realizacija osnovne i napredne pretrage. Opisana je stranica za dodavanje novog posta, kao i stranica na kojoj se vrši prikaz posta.

5. ZAKLJUČAK

Cilj ove teze je automatizacija procesa vođenja bloga. Kroz aplikaciju je omogućeno da bloger unosi nove postove na svoj blog bez potrebe za „ručnim“ manipulacijama u bazi podataka i generisanja HTML sadržaja prilikom unosa novog posta. Pri tome je potpuno automatizovan proces, kako dodavanja, tako i brisanja i ažuriranja postojećih postova. Pored toga, registrovani korisnici imaju mogućnost da čitaju postove blogera i da te postove lajkuju/dislajkuju i komentarišu. U aplikaciji je definisana i uloga administratora. On ima mogućnost izmene sadržaja kreiranih od strane drugih korisnika, ali može i da upravlja samim korisnicima. Administrator ima opciju da briše korisnike i menja im privilegije za korišćenje aplikacije.

Aplikacija je podložna daljim unapređenjima. Radi boljeg korisničkog iskustva, mogu se dodavati novi templejti za postove. Takođe, pored aploudovanja slika, korisno bi bilo i omogućiti korisnicima da aplouduju video sadržaj. Aplikacija se može unaprediti i omogućavanjem unosa novih kategorija od strane administratora. Za sva dalja unapređenja, treba oslušivati sugestije kako korisnika za vizuelni deo aplikacije, tako i blogera za dodavanje novih funkcionalnosti koje bi olakšale proces vođenja bloga.

LITERATURA

- [1] Internet programiranje, Aleksandra Smiljanić – <http://home.etf.rs/~aleksandra/IP.html>
- [2] JavaScript - *Sveobuhvati vodič*, David Flanagan (O'Reilly, prevod Mikroknjiga, 2008.)
- [3] PHP i MySQL - *Razvoj aplikacija za veb*, Luke Welling, Laura Thomson (O'Reilly, prevod Mikroknjiga, 2006.)
- [4] <http://php.net>
- [5] <https://jquery.com>
- [6] <https://www.w3schools.com>
- [7] <https://www.dev.mysql.com>
- [8] <http://www.draganmarkovic.net/cir/weblog.php?file=sta-je-html-i-htm.php>
- [9] Jovana Radojičić, "Veb prodavnica muzičkih instrumenata", diplomski rad, 2016.